Revisiting the Eliza classic in Prolog: example conversation

„I am unhappy.“
„How long have you been unhappy?“
„Six months. Can you help me?“
„What makes you think I help you?“
„You remind me of my sister.“
„Can you tell me more about your sister?“
„I like teasing my sister.“
„Does anyone else in your family like teasing your sister?“
„No. Only me.“
„Please go on.“

[The Art of Prolog, Sterling and Shapiro]

Revisiting the Eliza classic in Prolog: core “algorithm”

driven by stimulus-response patterns

I am <statement>.

How long have you been <statement>?

while the input is not „bye“

choose a stimulus-response pair

match the input to the stimulus

generate the reply from the response and the match

output the response

[The Art of Prolog, Sterling and Shapiro]

Revisiting the Eliza classic in Prolog: dictionary lookup

as association list for arbitrary keys:

```
lookup(Key,[(Key,Value)|Dict],Value).
lookup(Key,[(Key1,Value1)|Dict],Value) :-
  Key \= Key1,
  lookup(Key,Dict,Value).
```

will be used to store matches between stimulus and input

as binary tree for integer keys:

```
lookup2(Key,dict(Key,X,Left,Right),Value) :- !,
  X = Value.
lookup2(Key,dict(Key1,X,Left,Right),Value) :-
  Key < Key1,
  lookup2(Key,Left,Value).
lookup2(Key,dict(Key1,X,Left,Right),Value) :-
  Key > Key1,
  lookup2(Key,Right,Value).
```

[The Art of Prolog, Sterling and Shapiro]

Revisiting the Eliza classic in Prolog: representing stimulus/response patterns

numbered place-holder

numbered place-holder

```
pattern([i,am,1],['How',long,have,you,been,1,?]).
pattern([1,you,2,me],['What',makes,you,think,'I',2,you,?]).
pattern([i,like,1],['Does',anyone,else,in,your,family,like,1,?]).
pattern([i,feel,1],['Do',you,often,feel,that,way,?]).
pattern([1,X,2],['Please',you,tell,me,more,about,X]) :-
  important(X).
pattern([1],['Please',go,on,'.']).

important(father).
important(mother).
important(sister).
important(brother).
important(son).
important(daughter).
```

conditional pattern

[The Art of Prolog, Sterling and Shapiro]

Revisiting the Eliza classic in Prolog: main loop

```
reply([]) :- nl.
reply([Head|Tail]) :- write(Head),write(' '),reply(Tail).
```

```
eliza :- read(Input),
         eliza(Input),
         !.
eliza([bye]) :-
  writeln(['Goodbye. I hope I have helped you']).
eliza(Input) :-
  pattern(Stimulus,Response),
  match(Stimulus,Table,Input),
  match(Response,Table,Output),
  reply(Output),
  read(Input1),
  !,
  eliza(Input1).
```

find a Stimulus

match it with the Input, storing matches for place-holders in Table

substitute place-holders in Output

[The Art of Prolog, Sterling and Shapiro]

Revisiting the Eliza classic in Prolog: actual matching

```
match([N|Pattern],Table,Target) :-
  integer(N),
  lookup(N,Table,LeftTarget),
  append(LeftTarget,RightTarget,Target),
  match(Pattern,Table,RightTarget).
match([Word|Pattern],Table,[Word|Target]) :-
  atom(Word),
  match(Pattern,Table,Target).
match([],Table,[]).
```

place-holder

word

suppose D = [(a,b),(c,d)|X]

```
?- lookup(a,D,V)
V=b
?- lookup(c,D,e)
no
?- lookup(e,D,f)
yes
% D = [(a,b),(c,d),(e,f)|X]
```

The incomplete datastructure does not have to be initialized!

[The Art of Prolog, Sterling and Shapiro]